Sign Language Recognition Using Deep Learning

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering in Information Technology

By

Ms. Ankita Mallikarjun Dodamani	**19**
Mr. Amit Ravindra Kuveskar	**32**
Ms. Pranali Ratilal Wadile	**76**

Supervisor

Mrs. Brinzel Rodrigues

Assistant Professor

Department of Information Technology

St. John College of Engineering and Management

Vevoor Road, Palghar(East), 401404

UNIVERSITY OF MUMBAI

2021–2022

Sign Language Recognition Using Deep Learning

Submitted in partial fulfilment of the requirements

of the degree of

Bachelor of Engineering in Information Technology

By

Ms. Ankita Mallikarjun Dodamani	**19**
Mr. Amit Ravindra Kuveskar	**32**
Ms. Pranali Ratilal Wadile	**76**

Supervisor

Mrs. Brinzel Rodrigues

Assistant Professor

Department of Information Technology

St. John College of Engineering and Management

Vevoor Road, Palghar(East), 401404

UNIVERSITY OF MUMBAI

2021–2022

CERTIFICATE

This is to certify that the Project entitled "**Sign Language Recognition using Deep Learning**" is a bonafide work of **Dodamani Ankita Mallikarjun (PID No. EU1184012), Kuveskar Amit Ravindra (PID No. EU1164028), Wadile Pranali Ratilal (PID No. EU1154017)** submitted to University of Mumbai in partial fulfilment of the requirement for the award of the degree of "**Bachelors of Engineering in Information Technology**".

Mrs. Brinzel Rodrigues

Guide

Dr. Arun Saxena

Head of Department

Dr. G.V. Mulgund

Principal

B.E. Project Report Approval

This Project synopsis entitled ***Sign Language Recognition using Deep Learning*** by ***Ankita Mallikarjun Dodamani, Amit Ravindra Kuveskar, Pranali Ratilal Wadile*** is approved for the degree of ***Bachelors of Engineering in Information Technology*** from ***University of Mumbai***.

Examiners

1.----------

2.----------

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature
Ankita Mallikarjun Dodamani (EU1184012)

Signature
Amit Ravindra Kuveskar (EU1164028)

Signature
Pranali Ratilal Wadile (EU1154017)

Date:

ABSTRACT

According to the 2011 Census, In India, out of the total population of 121 crores, approximately 2.68 Crore humans are 'Disabled' (2.21% of the whole population)). Sign Language serves as a means for these people with special needs to communicate with others, but it is not a simple task. This barrier to communication has been addressed by researchers for years. The goal of this study is to demonstrate the MobileNets model's experimental performance on the TensorFlow platform when training the Sign language. Language Recognition Model, which can drastically reduce the amount of time it takes to learn a new language. Classification of Sign Language motions in terms of time and space Developing a portable solution for a real-time application. The Mobilenet V2 Model was trained for this purpose and an Accuracy of 70% was obtained.

Keywords: *Gesture Recognition, Deep Learning(DL), Sign Language Recognition(SLR), TensorFlow, Mobilenet V2.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Motivation:

Since the beginning of Evolution, Humans have kept evolving and adapting to their available surrounding. Senses have developed to a major extent. But Unfortunately there are some people who are born special. They are called special because they lack the ability to use all their five senses simultaneously.. According to WHO ,About 6.3% i.e. About 63 million people suffer from auditory loss in India. Research is still going on in this context. As per Census 2011, in India, out of the total population of 121 crore, about 2.68 Cr persons are 'Disabled' (2.21% of the total population). Out of 2.68 crore, 1.5 crore are males and 1.18 crore are females Majority (69%) of the disabled population resided in rural areas. There are various challenges faced by the specially abled people for Health Facilities, Access to Education, Employment Facilities and the Discrimination/ Social Exclusions tops it all. Sign Language is the language used to communicate with deaf people. These languages use visual-manual modality to convey meaning. Wherever communities of deaf people exists sign languages have developed as useful means of communication. Indian Sign Language Research and Training Centre is one such autonomous institute under ministry of social justice and empowerment which is dedicated to developing and upgrading the Indian Sign Language. Sign Language used Hand Gestures, Facial expressions and light bodily movements to convey the message. It is extremely important to understand and interpret the sign language and frame the meaningful sentence so as to convey the correct message which is extremely important and challenging at the same time

1.2 Problem System

To build a system that aids specially abled people, which will detect real time hand gestures and interpret its meaning that bridges the communication gap between the normal people and the specially-abled through the use of sign language.

1.3 Objective:

The objective of the project is to provide an efficient and accurate way to convert sign language into text or voice has aids for the hearing impaired for example, or enabling very young children to interact with computers (recognizing sign language), among others.

1.4 Scope:

The main purpose would be to accommodate a dialogue between signers and non-signers. This would be beneficial in emergency situations when there needs to be quick exchange of information like a conversation over the internet between a physician and his patient. The vocabulary can be extended with time as new words are added to expand the existing dataset. The System Scope is as follows:

- The system will be able to detect the Hand Gestures from Web Camera and display the corresponding Message.

CHAPTER 2

REVIEW OF LITERATURE

2.1 Smart Glove for Deaf And Dumb Patient

In the year 2018, P.B.Patel, Suchita Dhuppe, Vaishnavi Dhaye they represented smart glove for deaf and dumb patient. The glove is internally equipped with five flexsensors. For each specific gesture, the flex sensor produces a proportional change in resistance. The processing of these hand gestures is in Arduino uno Board which is an advance version of the microcontroller and the labview software. It compares the input signal with predefined voltage levels stored in memory. According to that required sound is produced which is stored is memory with the help of speaker.

2.2 Digital Text and Speech Synthesizer using Smart Glove for Deaf and Dumb

In the year 2017, Khushboo Kashyap, Amit Saxena, Harmeet Kaur, Abhishek Tandon, Keshav Mehrotra presented a system to increase the accuracy accelerometer was also incorporated which measured the orientation of the hand. It will be attached in the middle of the glove to measure the orientation of the glove. The output voltage of accelerometer changes depending on the orientation with respect to the earth.Unlike previous paper this model has 5 outputs from flex sensors and 3 from Accelerometer (the value of X,Y,Z axis). The controller used in our design is Arduino NANO. As all the outputs from both types of sensors are analog in nature, the output from five flex sensors and three outputs from the accelerometer are given as input to the analog inputs or analog port of the Arduino NANO. Arduino NANO has an inbuilt ADC in it which converts these analog inputs into digital output. After receiving these digital outputs coding is done accordingly. All the values of the flex sensors and of the accelerometer are mapped according to each gesture and then code is written for all the gestures. A bluetooth is also present in the hardware section. The Bluetooth module transmits the received data through a wireless channel which is then received by a Bluetooth receiver in the smartphone. Author had also developed a text to speech app using MIT App Inventor software which receives all the data and converts it into text or equivalent speech as per user‘s need. Android application proved to efficient in use. but the weight of gloves was still there.

2.3 Sign Language Recognitiom

In the year 2016, Anup Kumar, Karun Thankachan and Mevin M. Dominic presented a novel system to aid in communicating with those having vocal and hearing disabilities. It discusses an improved method for sign language recognition and conversion of speech to signs. The algorithm devised is capable of extracting signs from video sequences under minimally cluttered and dynamic background using skin color segmentation. It distinguishes between static and dynamic gestures and extracts the appropriate feature vector. These are classified using Support Vector Machines. Experimental results show satisfactory segmentation of signs under diverse backgrounds and relatively high accuracy in gesture and speech recognition.

2.4. Real-Time Recognition of Indian Sign Language

In the year 2019, represents a framework for a human computer interface capable of recognizing gestures from the Indian sign language. The complexity of Indian sign language recognition system increases due to the involvement of both the hands and also the overlapping of the hands. Alphabets and numbers have been recognized successfully. This system can be extended for words and sentences. Recognition is done with PCA (Principal Component analysis). This paper also proposes recognition with neural networks. Further it is proposed that number of finger tips and the distance of fingertips from the centroid of the hand can be used along with PCA for robustness and efficient results.

2.5 MobileNets for Flower Classification using TensorFlow

In the year 2017,Nitin R. Gavai, Yashashree A. Jakhade,Seema A. Tribhuvan, Rashmi Bhattad experimentally performed on MobileNets model on TensorFlow platform to retrain the flower category datasets, which can greatly minimize the time and space for flower classification compromising the accuracy slightly. is one of the pretrained models on the TensorFlow. Transfer learning a machine learning method which can use the existing knowledge learned from one environment and solve the new problem which is different but has some relation with the old one. Compared with the traditional neural network, it only needs to use a small quantity of data to train the model, and attain high exactitude with a short training time.

2.6 Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images

The paper presents results obtained by retraining and testing this sign language gestures dataset on a convolutional neural network model using Inception v3. The model consists of multiple convolution filter inputs that are processed on the same input. The validation accuracy obtained was above 90% This paper also reviews the various attempts that have been made at sign language detection using machine learning and depth data of images.

2.7 Indian Sign Language Gesture Recognition using Image Processing and Deep Learning

In this paper authors proposed a real time hand gesture recognition system based on the data captured by the Microsoft Kinect RGBD camera. Given that there is no one to one mapping between the pixels of the depth and the RGB camera, they used computer vision techniques like 3D construction and affine transformation. After achieving one to one mapping, segmentation of the hand gestures was done from the background noise. Convolutional Neural Networks (CNNs) were utilised for training 36 static gestures relating to Indian Sign Language (ISL) alphabets and numbers. The model achieved an accuracy of 98.81% on training using 45,000 RGB images and 45,000 depth images. Further Convolutional LSTMs were used for training 10 ISL dynamic word gestures and an accuracy of 99.08% was obtained by training 1080 videos. The model showed accurate real time performance on prediction of ISL static gestures.

2.8. Sign Language Recognition

In this paper, an image processing algorithm is presented for the interpretation of the Taiwanese Sign Language, which is one of the sign languages used by the majority of the deaf community. The process involves two-layer classifications. At first, coarse classification is done according to detection of hand motion and tracking the hand location and second classification is based on key frame selection and hand shape recognition of key frames. Motion history image and Fourier descriptor are used for motion direction recognition and key frame selection respectively. Generic cosine descriptor (GCD) has been proposed for feature extraction of hand postures. GCD is invariant to scale, translation and rotation of hand shapes.

2.9 Signet: A Deep Learning based Indian Sign Language Recognition System

In this paper, author proposed a signer independent deep learning-based methodology for building an Indian Sign Language (ISL) static alphabet recognition system they also reviewed various existing methods in sign language recognition and implement a Convolutional Neural Network (CNN) architecture for ISL static alphabet recognition from the binary silhouette of signer hand region. They also discuss in detail, the dataset used along with the training phase and testing phase of CNN. The proposed method was successfully implemented an accuracy of 98.64% which is better than most of the currently existing methods.

2.10 Static Sign Language Recognition Using Deep Learning

This system is based on a skin-color modelling technique, i.e., explicit skin-color space thresholding. The skin-color range is predetermined that will extract pixels (hand) from non–pixels (background). The images were fed into the model called the Convolutional Neural Network (CNN) for classification of images. Keras was used for training of images. Provided with proper lighting condition and a uniform background, the system acquired an average testing accuracy of 93.67%, of which 90.04% was attributed to ASL alphabet recognition, 93.44% for number recognition and 97.52% for static word recognition, thus surpassing that of other related studies.

2.11 Deep Learning-Based Approach for Sign Language Gesture Recognition With Efficient Hand Gesture Representation

The proposed system is evaluated on a very challenging dataset, which consists of 40 dynamic hand gestures performed by 40 subjects in an uncontrolled environment. The results show that the proposed system outperforms state-of-the-art approaches, demonstrating its effectiveness. A 3D convolutional neural network (3DCNN) has been used for video modeling, which is an extended version of standard CNNs that uses spatiotemporal filters. The architecture has been explored previously in several video analysis fields for spatiotemporal feature representation. Instead of training a 3DCNN from scratch, using domain adaptation on pretrained instances is preferred.

2.12 Common Garbage Classification Using MobileNet

This study used MobileNet to generate a model that classifies common trash according to the following categories: glass, paper, cardboard, plastic, metal, and other trash. The model was built using transfer learning from a model trained on the ImageNet Large Visual Recognition Challenge dataset. The image classifier was trained using Python and TensorFlow. The resulting baseline model, with a final test accuracy of 87.2% was optimized and quantized. In the Andoid app development, the optimized model (with 89.34% confidence) is preferred over the quantized model (with 1.47% confidence) based on the test using a plastic image. The model app was successfully installed in a Samsung Galaxy S6 Edge+ mobile phone. The installed mobile app successfully identified a cardboard material in an image with a cardboard container. An Android app based on the classifier model was created using Android Studio 3.0.1. The app installed on mobile devices with Tensorflow Mobile.

2.13. Sign Language Recognition System using TensorFlow Object Detection API

In this paper, we propose a method to create an Indian Sign Language dataset using a webcam and then using transfer learning, train a TensorFlow model to create a real-time Sign Language Recognition system. The proposed system is designed to develop a real-time sign language detector using a TensorFlow object detection API and train it through transfer learning for the created dataset. For data acquisition, images are captured by a webcam using Python and OpenCV. The system achieves a good level of accuracy even with a limited size dataset.

2.14. Sign Language Recognition

The proposed system does not involve any complex like a glove and is purely a vision based system where a user need not wear any type of cumbersome components for the recognition purpose. The system involves a web camera which is used to capture the image of the hand a processor for classification and recognition purpose and an output unit which can be a speaker. A database is created for the classification of the hand gestures taken through the web camera. The database may consist of several numbers of poses associated with the same gesture. The ANN method used here is a supervised learning method. This method involves pattern

recognition. The patterns of the hand have to be recognized in order to identify the sign. The static gestures are used here for the recognition process.

2.15. Sign Language Recognition Techniques- A Review

In this Paper have implemented Convolutional neural networks (CNN) for training 36 static gestures that are related to Indian sign language including alphabets and numbers. CNN is used for the classification and recognition. The first step is preprocessing that does the elimination of face to only retain the hand pixels. Proposed methodology that consists of three phases. Training phase, testing phase and recognition phase. The pre-processing has two steps segmentation and filtering, Global thresholding method is used for segmentation. They have done recognition for gestures. The dataset includes 720 images. Combinational parameters of Hu invariant moment and structural shape descriptor, gives another feature vector that has to be extracted from the image. Training is done through Multiclass Support Vector Machine (MSVM). The segmentation step was performed on the image to detect the shape of the gesture. The detected region was then transformed into a binary image. After obtaining a binary image the Euclidean distance transformation was applied on that image.

Table 2.1: Literature Survey

Sr. No	Title	Journal	Conclusion	Research Gap
1	Smart Glove for Deaf And Dumb Patient	IJARSE, Volume No.07, Special Issue No. 03, April 2018	This Method uses flex sensors equipped on gloves and detects the fluctuation in voltage and converts it to audio output using speakers	Heavy to Carry gloves. All the signs were not converted into sound. Delay in result due to fluctuations in resistance.
2	Digital Text and Speech Synthesizer using Smart Glove for Deaf and Dumb	IJARECE, Volume 6, Issue 5, May 2017	The main objective of this project is to convert basic commands used in day to day life. Use of android Application made it efficient for use.	The flex sensors gave less accurate results for gestures with high deflection values
3	Sign Language Recognition	3rd InCI Conf. on Recent Advances in Information Technology	It follows an image-based approach. using two layer classifier: GCD Further areas of improvement such as increasing the system performance under robust and unfavourable environment	It couldn't Recognize more gesture (like those involving two hands or facial cues)

Sr. No	Title	Journal	Conclusion	Research Gap
4	Real-Time Recognition of Indian Sign Language	ICCIDS-2019	This FCM based real-time sign language recognition system, for recognising the words of Indian Sign Language has produced 75 % accuracy in gesture labelling and this is somewhat higher than the similar systems. Also, the developed system is much better than other systems, since it is capable of recognising 40 words of ISL in real-time while the similar systems have the capability to recognize static gestures only.	Extend the system by combining Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to capture the spatial and temporal features and more words will be added to the system.
5	MobileNets for Flower Classification using TensorFlow	International Conference on Big Data, IoT and Data Science (BID) Vishwakarma Institute of Technology	In the traditional flower classification methods, convolutional neural network uses multilayer convolution to extract features and combine them automatically. TensorFlow has advantages of high accessibility, high flexibility. Compared with the traditional neural network, it only needs to use a small quantity of data to train the model	TensorFlow compromises the accuracy.
6	Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images	IEEE 2018, International Conference on Smart City and Emerging Technology (ICSCET)	The model consists of multiple convolution filter inputs that are processed on the same input. The validation accuracy obtained was above 90% This paper also reviews the various attempts that have been made at sign language detection using machine learning and depth data of images.	Further training the neural network to efficiently recognize symbols involving two hands. Improving the results by using linear classifiers, Including dynamic hand gestures in addition to the current static finger spelling
7	Indian Sign Language Gesture Recognition using Image Processing and Deep Learning	IEEE 2019	This paper proposes a real-time model for ISL gesture recognition, based on the incoming image data from the Kinect. Effective real time background subtraction was done using depth perception	To be focused on real-time prediction of more words related to ISL and also on sentence formation

Sr. No	Title	Journal	Conclusion	Research Gap
			techniques. Computer vision techniques were used to achieve one-to-one mapping between the depth and the RGB pixels.	
8	Sign Language Recognition	2012 1st International Conference on Emerging Technology Trends in Electronics, Communication and Networking	The complexity of Indian Sign Language increases with the involvements of two hands. The system was successful to recognize alphabets and numbers using one hand. .	This paper proposes that the recognition can be more robust using Neural Networks
9	Signet: A Deep Learning based Indian Sign Language Recognition System	International Conference on Communication and Signal Processing, April 4-6, 2019	A vision based deep learning architecture for signer independent Indian sign language recognition system. The system was successfully trained on all 24 ISL static alphabets with a training accuracy of 99.93% and with testing and validation accuracy of 98.64%.	Additional modules and techniques to form a fully automated sign language recognition system in the future. Facial expression and context analysis are the other part to be included in sign language recognition.
10	Static Sign Language Recognition Using Deep Learning	International Journal of Machine Learning and Computing, Vol. 9, No. 6, December 2019	System was able to achieve 99% training and testing accuracy of 90.04% in letter recognition, 93.44% in number recognition and 97.52% in static word recognition, obtaining an average of 93.667% based on the gesture recognition with limited time. Each system was trained using 2,400, 50 × 50 images of each letter/number/word gesture	Recognition can be done using aid such as gloves or hand markings.
11	Deep Learning Based Approach for Sign Language Gesture Recognition With Efficient Hand Gesture Representation	IEEE Access, October 2020	A robust face detection algorithm and the body parts ratios theory were utilized for gesture space estimation and normalization. Two 3DCNN instances were used separately for learning the ne-grained features of the hand shape and the coarse-grained	Strategies for temporal aspect modelling and also test the system for Real-time hand gesture recognition

Sr. No	Title	Journal	Conclusion	Research Gap
			features of the global body configuration.	
12	Common Garbage Classification Using MobileNet	IEEE Conference 2018	The model app in the Android phone used was able to identify a cardboard material in an image with a cardboard container. The model's confidence value in the said test was 99.61%.	Rerun the training using more steps as this may improve the quantized model performance since a quantized model is fit for mobile devices than models with no quantization.
13	Sign Language Recognition System using TensorFlow Object Detection API	International Conference on Advanced Network Technologies and Intelligent Computing (ANTIC-2021), part of the book series 'Communications in Computer and Information Science (CCIS)', Springer.	The system detects sign language in real-time. For data acquisition, images have been captured by a webcam using Python and OpenCV which makes the cost cheaper. The developed system is showing an average confidence rate of 85.45%.	Dataset can be enlarged so that the system can recognize more gestures.
14	Sign Language Recognition System	IEEE 2nd International Conference, ICIIES'15	The system is robust under various lighting conditions. Image is taken & the pre-processing & morphological operations performed. A database of 25 images is created for 5 types of Gestures	
15	Sign Language Recognition Techniques- A Review	2020 IEEE International Conference for Innovation in Technology (INOCON) Bengaluru, India. Nov 6-8, 2020	Explains the methods of Sign language recognition and describes the steps involved in gesture recognition which include acquisition, segmentation, feature extraction till recognition and classification.	Dataset can be enlarged so that the system can recognize more gestures.

CHAPTER 3

SYSTEM DESIGN

3.1 Functional Requirements

- Capturing the Sign Language Gestures from Web Camera.
- Recognizing the Appropriate meaning of the Gesture.

3.2 Non Functional Requirements

- Availability: All the system who can run Colab on chrome.
- Performance: based on 10gb cloud GPU

3.3 Specific Requirements

3.3.1 Hardware Requirements

1.	System	:	Intel i5 (or equivalent) processor or later
2.	RAM	:	8 GB (minimum)
3.	Disk Drive	:	1 TB HDD / SSD
4.	Web Camera	:	720*1280 minimum
5.	Graphics Processor	:	NVIDIA

3.3.2 Software Requirements

1. OS : Windows 10
2. Google Colab
3. Python Environment 3.6 or later
4. Libraries : Tensorflow, Cv2, Numpy, Matplotlib, Time, Sklearn
5. Internet Browser

3.4 Use Case Diagrams and Description

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. Use case diagrams are also known as behaviour diagrams, and they are used to depict a set of behaviours that some systems should or should do in

coordination with one or more external users (actors). every user should deliver some measurable and valuable output. Use case diagrams are both behaviour and structural diagrams, as they represent the system's activity. They are a specific example of class diagrams in which classifiers are restricted to either actors or use cases associated to one another using association.

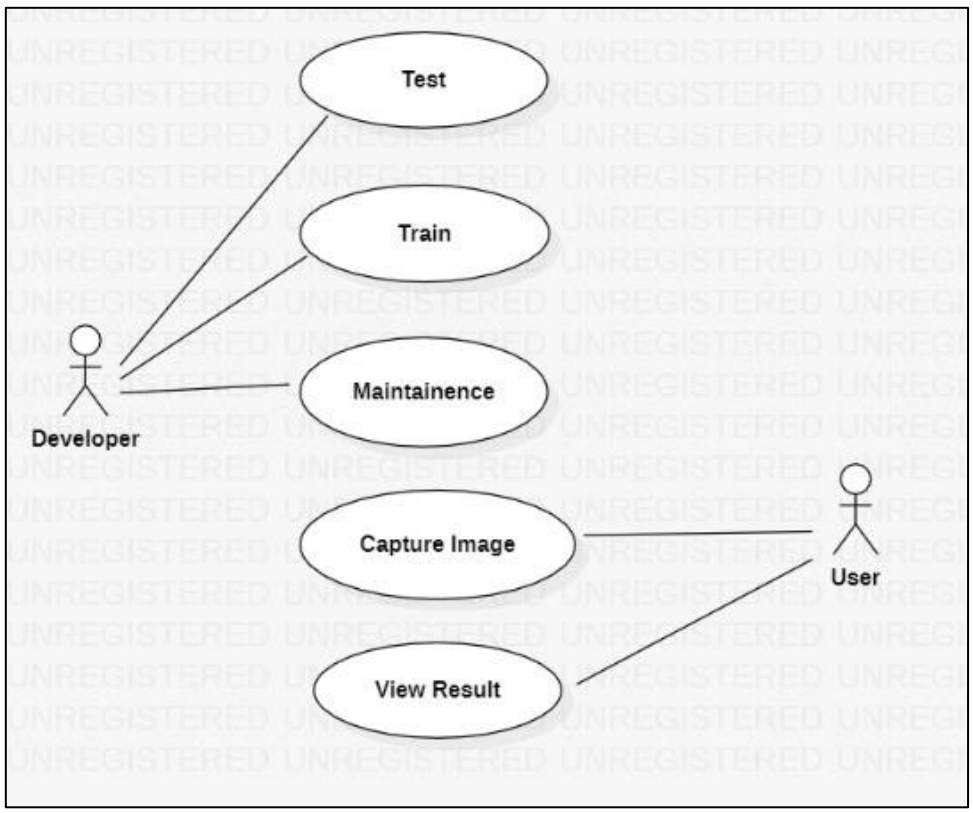

Figure 3. 1: Use Case Diagram

Use case diagrams helps to specify the required usage of a system under design or analysis (subject) -what the system is supposed to do, The functionality offered by the subject - what the system can do and the Requirements the specified subject poses on its environment - by defining how environment should interact with the subject so that it will be able perform its services.

Use Case description for “Capture Image”

Table 3.4. 1:Use Case Description for Capture Image

Use Case	Image Capturing from Real time Camera
Primary Actor	Webcam
Goal in Context	Webcam captures live feed
Pre Conditions	User’s System must have an active webcam connection
Trigger	On capturing live feed, movements should be tracked
Scenario	Live feed is captured
Priority	Every frame captured is sent for processing
Secondary Actor	User
Exception	The Software breaks

Use Case description for “View Result”

Table 3.4. 2:Use Case Description for View Result

Use Case	Display gesture meaning
Primary Actor	Computer/System
Goal in Context	Display the meaning according to gesture recognition
Pre Conditions	User’s System must have an active webcam connection
Trigger	The sign is recognized
Scenario	Images are passed to the network for recognition
Priority	Every gesture must be recognized
Secondary Actor	User
Exception	Gesture cannot be recognized by the software

CHAPTER 4

ANALYSIS MODELLING

4.1 Activity Diagram

Activity diagram is a behavioral diagram which depicts the behavior of the system. It portrays the control flow from a start point to a finish point showing the various decision paths that exists while the activity is being executed.

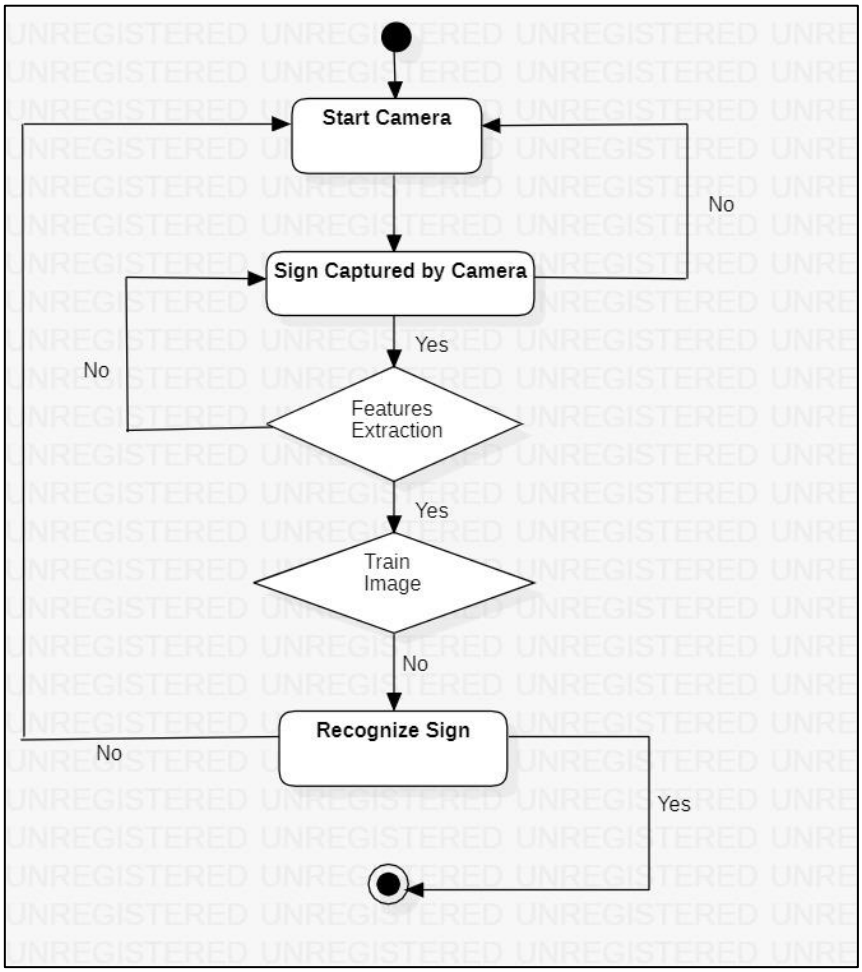

Figure 4. 1: Activity Diagram

4.2 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

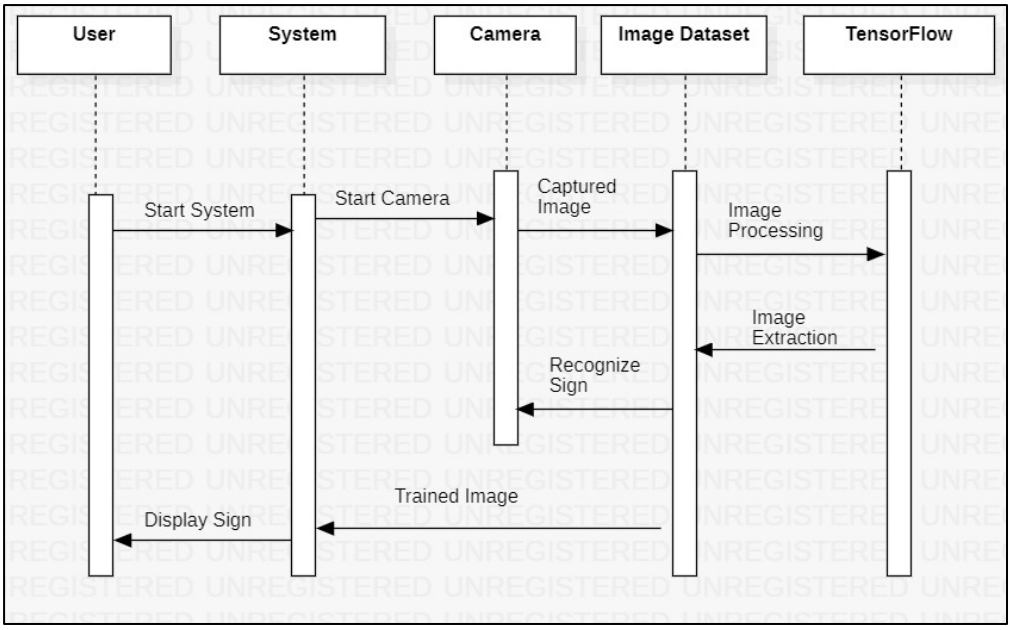

Figure 4. 2:Sequence Diagram

4.3 Functional Modeling

Data flow diagrams provides a graphical representation of how information moves between processes in a system. A Data Flow Diagram shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

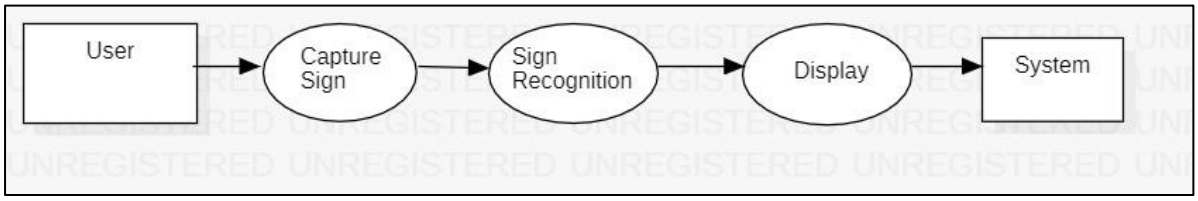

Figure 4. 3: DFD Level 0 Diagram

In this level 0 data flow diagram, the whole system is represented with the help of input, processing and output. The input to the gesture recognition system is the live feed from the camera which contains the gestures performed by the user. The camera provides the frames which can be mapped to their corresponding meanings.

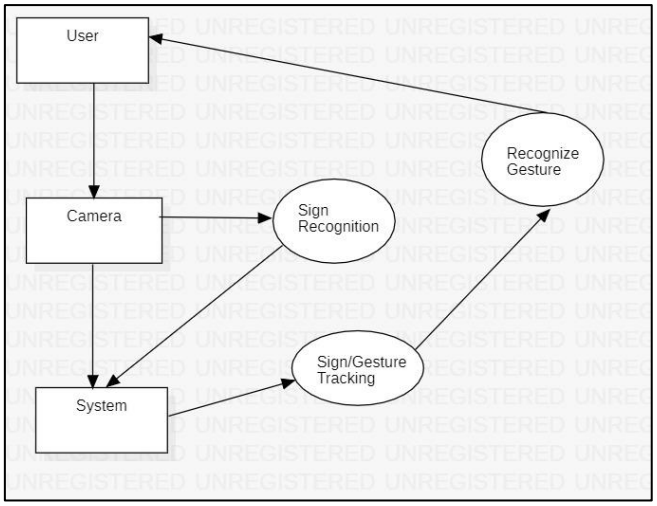

Figure 4. 4: DFD Level 1 Diagram

In level 1 data flow diagram, the gesture recognition module is explained in further detail. The camera provides live feed of the user actions. Operations are performed to enhance the hand movements. The hand movement is recognized and sent to the gesture tracking module. The gesture tracking module checks for the gesture in the pre trained network of gestures and their respective meanings. The gesture control module maps the gesture to its meaning. Thus in these processes the gesture is recognized and meaning is displayed.

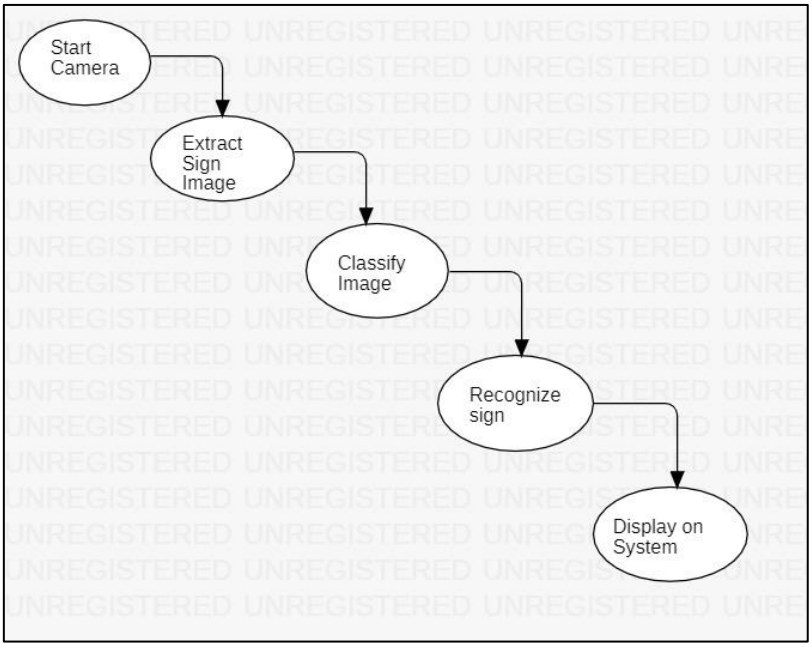

Figure 4. 5: DFD Level 2 Diagram

4.4 Time Line Chart

Time line charts illustrate events in chronological order the progress of a project, advertising campaign and acquisition process in whatever unit of time the data was recorded: week, month and a year

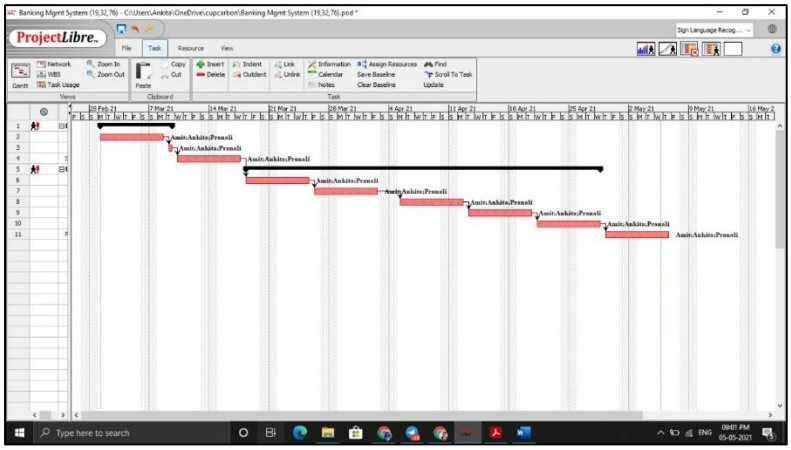

Figure 4. 6: Timeline Chart

4.5 WBS Chart

Time line charts illustrate events in chronological order the progress of a project, advertising campaign and acquisition process in whatever unit of time the data was recorded: week, month and a year

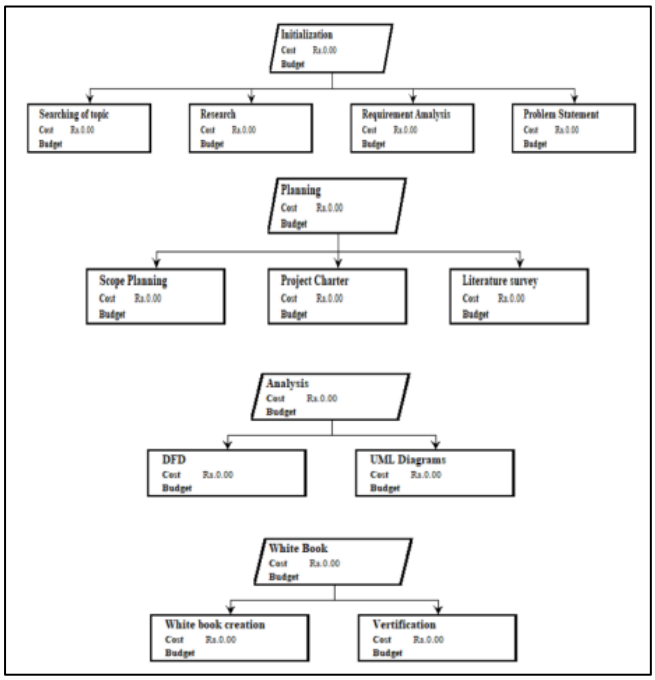

Figure 4. 7:WBS Char

CHAPTER 5

PROPOSED SYSTEM

5.1 Scope

The experimental setup for the Sign Language Recognition model utilizing MobileNet on the TensorFlow framework is covered in this section. The categorization model is divided into the four stages below: Phases include image preprocessing, training, verification, and testing. For our project we have made our data set of around 15,000 images which consist of 26 Alphabets(A to Z), Numerals (0-9).After collecting all the images we labeled them using LabelImg[18] The Images collected are then divided into two categories: Train and Test. Train dataset is used to train the model and the Verification Phase uses the Test dataset to verify the accuracy. Then the Model is used to test the model in Real-Time. In terms of our project's scope, our major goal is to create a model that can recognize the numerous signs that define Letters, Numerals, and Gestures using mobilenets. Using the Object Detection Technique, the Trained model can recognize the indicators in real-time. The idea behind this project is to develop an application that is handy and can detect the hand gestures (signs) and recognize what the specially-abled person is trying to speak with a motive to help to ease the efforts required by the specially-abled people to communicate with other Normal People.

5.2 Architecture Diagram

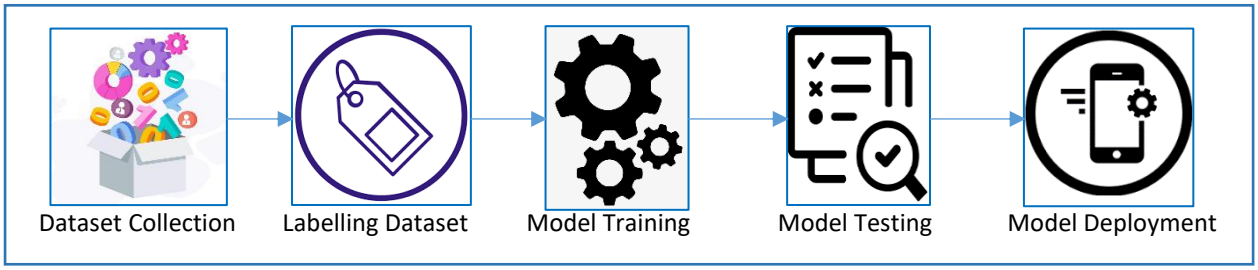

Figure 5.2.1: Block Diagram of Model for Sign Language Recognition (SLR)

The proposed system is divided in various phases. The first phase is the system Installation and Setup we had done for the project. The very basic & most important task was to create a separate Virtual environment, so that all the installations remain safe and isolated from other dependencies. After activating the virtual environment we now installed the required dependencies and added a virtual environment to python kernel. The first phase is completed here. Second phase is the dataset collection. In this phase we carry out Image Collection & Labelling Process. This phase has various sub phases.

- Importing necessary dependencies like uuid for generating random ids, CV2 which is Open CV python open source library which is used for computer vision in AI,ML, Face recognition etc, OS library which helps us to navigate through directiories and provides a way to deal with file .Name path and directories, time module allows us to work with time in python.
- Defining the Images: Here we predefine the images labels. In our project we have A to Z letters and 0 to 9 numerals which are on mapped in a list since we have to deal with many folders and large data sets it becomes important setup a well defined folders structure. Next phase deals with defining correct folder and structure and OS module is used here. After all the settlements we now capture images and create the dataset for our project using OpenCv library and webcam. After creating the dataset we now need to label them. We found the LabelImg library on github using which all the images are labelled. pyQt5,a GUI toolkit was necessary for labelImg was installed. Labelling an XML file was generated as reason of labelling the image. This step was very tricky as even a single missing XMl file used to throw errors in further steps. Every image had to have a PNG/JPG file along with atleast one XML file. The next phase of the proposed system was to manually divide the collected images in two folders train and test.
- The next Phase is the training phase. We had use Tensorflow , Mobilenet V2 Model which gives a comparatively better results. When compared with Googles net model and VGG16, it was concluded that it gave fairly similar results but the no of parameters required for the purpose was significantly very less , which makes it ideal to use. A URL downloader that helped us to download files directly from server.
- In next step, we cloned the Tensorflow models, library from Github and installed Tensorflow object detection API & Tensorflow upgrade.protobut, a library used for purpose of writing proto description of data structure. We wish to store, it creates and implements automatic encoding & parsing of protocol bufferdata with an efficient binary format. All the corresponding dependencies were also installed after the installation was completed the next part is to verify if all the packages were successfully installed. Libraries like matplotlib for data visualization was installed.
- After all the packages were installed with correct compatible versions, next step was to create a labelmap. Here each label corresponds to unique ID. Now all the XML files were converted for tensorflow object detection API. There were two records formed at

the end of this step, train.record & test.record copied to training location. Now this configuration was updated for transfer learning process.

- After the configurations was updated according to our requirements, the model was now trained.

  Steps:

  1.Model_main_tf2 was run.

  2.Pipeline config file & Retrained config file both were passed as a parameter to train. Since the training was taking a lot of time TensorFlow GPU was installed
- The Trained Model is now completed and ready to be tested.The model was tested Real time using Webcamera.
- In the next step, we calculated the Model performance. The F1 score is used to test the system's performance. In relation to a particular positive class, the F1 score blends precision and recall. The F1 score is a measured average of precision and recall, with 1 representing the best and 0 representing the worst. In the most simple terms, higher F1 scores are generally better.

  The F1 score evaluated for the proposed system is; 0.7959, F1 score: 79.59%

CHAPTER 6

TECHNOLOGIES USED

6.1.Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it. Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, games, and desktop apps.

6.2 Jupyter notebook

The Jupyter Notebook is an open-source web application that allows data scientists to create and share documents that integrate live code, equations, computational output, visualizations, and other multimedia resources, along with explanatory text in a single document. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python, and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python.Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The “notebook” term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context.

6.3 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very

important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features. It is an open-source computer vision and machine learning software library developed by Intel. It provides a common infrastructure for applications related to computer vision and its associated fields. It is used to speed up the use of real-time machine recognition of images, objects, and video processing applications.

6.4 UUID

Universally Unique Identifiers, or UUIDS, are 128 bit numbers, composed of 16 octets and represented as 32 base-16 characters, that can be used to identify information across a computer system. This specification was originally created by Microsoft and standardized by both the IETF and ITU.UUIDs are generally used for identifying information that needs to be unique within a system or network thereof. Their uniqueness and low probability in being repeated makes them useful for being associative keys in databases and identifiers for physical hardware within an organization. One of the benefits to UUIDs is that they don't need to be issued by a central authority, but can be generated independently and then used across a given system without suspicion that a duplicate. or colliding, UUID has been generated elsewhere. Apple, Microsoft, Samsung, and others use UUIDs, either defined by the IETF spec or a proprietary variant, to identify and track hardware both internally and sold to consumers.There are 5 different versions of UUIDs, excluding the Nil UUID version, which is a special case UUID where all its bytes are set to 0, and most contain some variants that allow for special cases specific to vendors like Microsoft.

6.5 OS

The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The os and os.path modules include many functions to interact with the file system. Python OS module provides the facility to establish the

interaction between the user and the operating system. It offers many useful OS functions that are used to perform OS-based tasks and get related information about operating system.

6.6 Time

As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. So before starting with this module we need to import it. Importing time module The time module comes with Python's standard utility module, so there is no need to install it externally. We can simply import it using the import statement.

6.7 LabelImg

LabelImg is an image annotation tool which is quite simple and easy-to-use for the beginner or non-technical guys who are concerning about how to do computer vision projects. LabelImg is a graphical image annotation tool which provides images with bounding boxes after labeling. It is written in Python and uses Qt (one of the most common GUI for python) for its graphical interface. The output labeled data are saved as XML files in PASCAL VOC format, the format used by ImageNet. It is open-source then it is free to use. It has both online and offline versions. The offline interface is user-friendly and allows the user to make fast annotations. The output annotation images are saved separately as XML files along with Pascal VOC format or YOLO format. These are common formats supported by many ML packages. It can be run in multiple operating systems such as Linux, Mac or Windows.

6.8 PyQt5

PyQt5 is implemented as a set of Python modules. It has over 620 classes and 6000 functions and methods. It is a multiplatform toolkit which runs on all major operating systems, including Unix, Windows, and Mac OS. PyQt5 is dual licensed. PyQt5 is cross-platform GUI toolkit, a set of python bindings for Qt v5. One can develop an interactive desktop application with so much ease because of the tools and simplicity provided by this library. A GUI application consists of Front-end and Back-end. PyQt5 has provided a tool called 'QtDesigner' to design the front-end by drag and drop method so that development can become faster and one can give more time on back-end stuff.

6.9 TensorFlow

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. TensorFlow is an open source machine learning framework for all developers. It is used for implementing machine learning and deep learning applications. To develop and research on fascinating ideas on artificial intelligence, Google team created TensorFlow. TensorFlow is designed in Python programming language, hence it is considered an easy to understand framework. It is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

6.10 Protobuf

Protocol buffers (Protobuf) are a language-agnostic data serialization format developed by Google. Protobuf is great for the following reasons: Low data volume: Protobuf makes use of a binary format, which is more compact than other formats such as JSON. Persistence: Protobuf serialization is backward-compatible. This means that you can always restore previous data, even if the interfaces have changed in the meantime. Design by contract: Protobuf requires the specification of messages using explicit identifiers and types. Requirement for gRPC: gRPC (gRPC Remote Procedure Call) is an efficient remote procedure call system that makes use of the Protobuf format.

6.11 Wget

Wget is a free twenty-five-year-old command-line program that can retrieve files from web services using HTTP, HTTPS, and FTP. If you use it with Python, you're virtually unlimited in what you can download and scrape from the web. Wget can recover from broken transfers, making it a good solution for downloading files over unstable or slow networks. Wget uses the Range HTTP header to continue a download from where it left off until the whole file is received. Wget is a URL network downloader that can work in the background, and it helps in downloading files directly from the main server. In Python, this task is done by using the wget module.

6.12 Artificial Intelligence (AI)

AI is the ability of a machine to display human-like capabilities such as reasoning, learning, planning and creativity. AI enables technical systems to perceive their environment, deal with what they perceive, solve problems and act to achieve a specific goal. The computer receives data - already prepared or gathered through its own sensors such as a camera - processes it and responds. AI systems are capable of adapting their behaviour to a certain degree by analysing the effects of previous actions and working autonomously. Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.

6.13 Deep Learning

Deep learning requires large amounts of labeled data. For example, driverless car development requires millions of images and thousands of hours of video. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less. Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected. The more deep learning algorithms learn, the better they perform.

6.14 MobileNetV2

MobileNetV2 is a very effective feature extractor for object detection and segmentation. For example, for detection when paired with the newly introduced SSDLite [2] the new model is about 35% faster with the same accuracy than MobileNetV1. We have open sourced the model under the Tensorflow Object Detection API. MobileNetV2 is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers.MobileNet-v2 is a convolutional neural network that is 53 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals.

CHAPTER 7

IMPLEMENTATION

7.1 Model Architecture

7.1.1. Convolution Neural Network

The Convolution Neural Network (CNN) is a deep learning method inspired by human neurons. A neural network is a collection of artificial neurons known as nodes in technical terms. A neuron in simple terms is a graphical representation of a numeric value. These neurons are connected using weights(numerical values).

Training refers to the process where a neural network learns the pattern required for performing the task such as classification, recognition, etc. When a neural network learns, the weight between neurons changes which results in a change in the strength of the connection as well. A typical neural network is made up of various levels. The first layer is called the input layer, while the output layer is the last. In our case of recognizing the image, this last layer consists of nodes that represent a different class. We have trained the model to recognize Alphabets A to Z & Numerals 0 to 9. The likelihood of the image being mapped to the class represented by the node is given by the output neuron's value.

Generally, there are 4 layers in CNN Architecture: the convolutional layer, the pooling layer, the ReLU correction layer, and the fully-connected layer. The Convolutional Layer is CNN's first layer, and it works to detect a variety of features. Images are fed into the convolutional layer, which calculates the convolution of each image with each filter. The filters match the features we're looking for in the photographs to a match. A feature map is created for each pair (picture, filter). The pooling layer is the following tier. It takes a variety of feature maps as inputs and applies the pooling method to each of them individually. In simple terms, the pooling technique aids in image size reduction while maintaining critical attributes. The output has the same number of feature maps as the input, but they are smaller. It aids in increasing efficiency. and prevents over-learning. The ReLU correction layer is responsible for replacing any negative input values with zero. It serves as a mode of activation. The fully connected layer acts as the final layer. It returns a vector with the same size as the number of classes the image must be identified from.

7.1.2. Mobilenet

MobileNet is a class of CNN that was open-sourced by Google, and therefore, this gives us an excellent starting point for training our classifiers that are insanely small and insanely fast. It is a CNN Architecture that is faster as well as a smaller model. It makes use of a Convolutional layer called depth-wise separable convolution.

Table 7.1.2. 1:MobileNet parameter and accuracy comparison against GoogleNet and VGG 16

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

MobileNets for TensorFlow are a series of mobile-first computer vision models that are designed to maximize accuracy while taking into account the limited resources available for an on-device or embedded application. As seen in the table at the Right, it can be concluded that Mobile net gives fairly similar results as compared with Google Net Model and VGG 16, but the number of Parameters required for the purpose is significantly less, which makes it ideal to use. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers. In MobileNetV2, there are two types of blocks. One is residual block with stride of 1. Another one is block with stride of 2 for downsizing. There are 3 layers for both types of blocks. This time, the first layer is 1×1 convolution with ReLU6. The second layer is the depthwise convolution. The third layer is another 1×1 convolution but without any non-linearity. It is claimed that if ReLU is used again, the deep networks only have the power of a linear classifier on the non-zero volume part of the output domain. And there is an expansion factor t. And t=6 for all main experiments. If the input got 64 channels, the internal output would get 64×t=64×6=384 channels. The first layer of the MobileNet is a full convolution, while all following layers are Depthwise Separable Convolutional layers. All the layers are followed by batch normalization and ReLU activations. The final classification layer has a softmax activation.

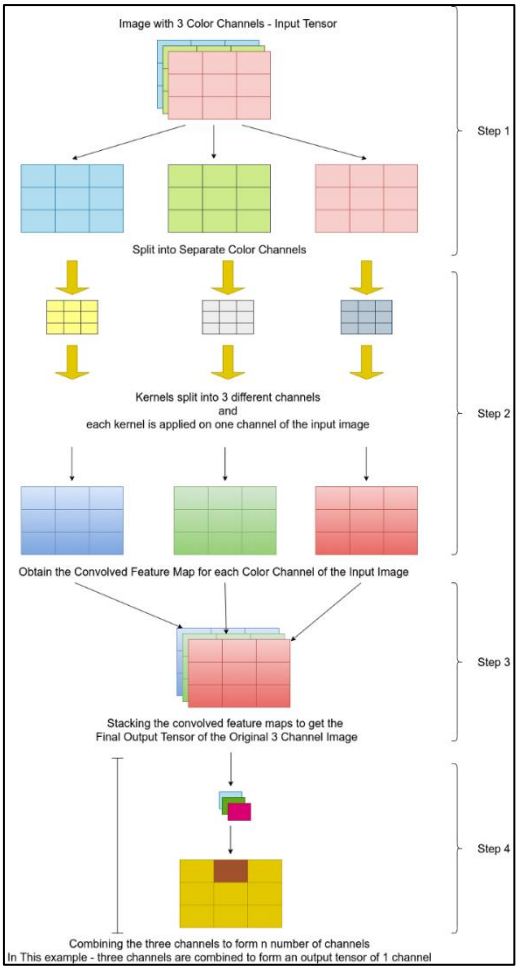

Figure 7.1.2. 1:Diagramatic explanation of Depth Wise Separable Convolution

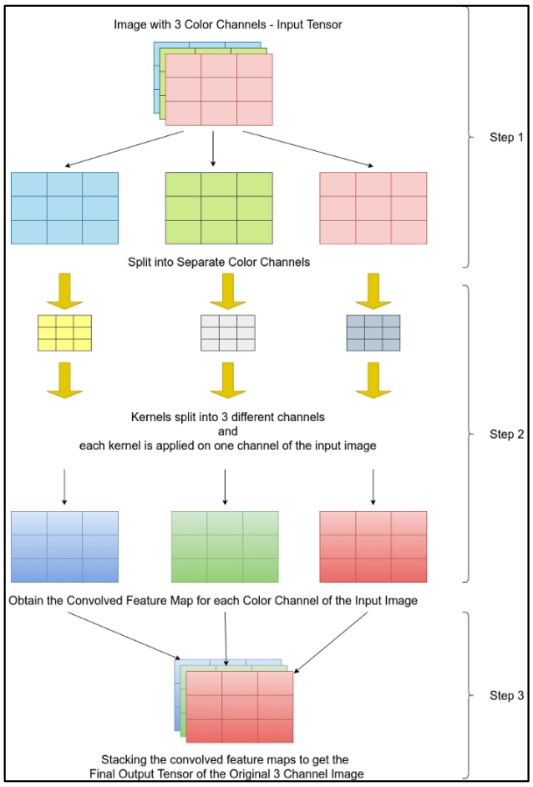

Figure 7.1.2.2: Depth wise Convolutions

The main difference between the 2D convolutions in CNN and Depthwise convolutions is that the 2D Convolutions are performed over multiple channels, whereas in Depthwise convolutions each channel is kept separate. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity.

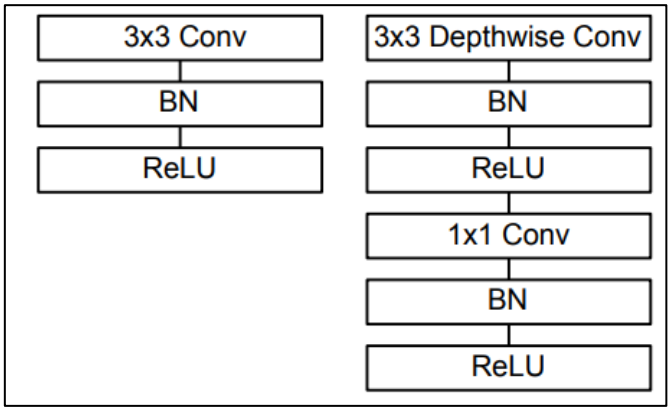

Figure 7.1.2.4: . Left: Standard Convolutional layer, Right: Depthwise Separable Convolutional layers in MobileNet.

7.2 Working of the Project

7.2.1 Image Collection Pseudo Code

```
for label in labels:
    cap = cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(number_imgs):
        print('Collecting image {}'.format(imgnum))
        ret, frame = cap.read()
        imgname =
os.path.join(IMAGES_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame', frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
cap.release()
cv2.destroyAllWindows()
```

7.2.2 Setting Paths

```
import os

CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME =
'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet
_v2_fpnlite_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'


paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-
trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'export'),
```

```
    'TFJS_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow',
'workspace','models',CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }

files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models',
CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'],
TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'],
LABEL_MAP_NAME)
}

for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
```

7.2.3. Downloading TF Models Pretrained Models from Tensorflow Model Zoo and Installing TFOD

```
# https://www.tensorflow.org/install/source_windows

if os.name=='nt':
    !pip install wget
    import wget

if not os.path.exists(os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection')):
    !git clone https://github.com/tensorflow/models {paths['APIMODEL_PATH']}
# Install Tensorflow Object Detection
if os.name=='posix':
    !apt-get install protobuf-compiler
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --
python_out=. && cp object_detection/packages/tf2/setup.py . && python -m pip
install .

if os.name=='nt':
    url="https://github.com/protocolbuffers/protobuf/releases/download/v3.15.6/protoc-
    3.15.6-win64.zip"
    wget.download(url)
    !move protoc-3.15.6-win64.zip {paths['PROTOC_PATH']}
     !cd {paths['PROTOC_PATH']} && tar -xf protoc-3.15.6-win64.zip
```

```
    os.environ['PATH'] += os.pathsep +
    os.path.abspath(os.path.join(paths['PROTOC_PATH'], 'bin'))
    !cd Tensorflow/models/research && protoc object_detection/protos/*.proto --
    python_out=. && copy object_detection\\packages\\tf2\\setup.py setup.py &&
    python setup.py build && python setup.py install
    !cd Tensorflow/models/research/slim && pip install -e .

VERIFICATION_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection', 'builders', 'model_builder_tf2_test.py')
# Verify Installation
!python {VERIFICATION_SCRIPT}

!pip install tensorflow –upgrade
!pip uninstall protobuf matplotlib -y
!pip install protobuf matplotlib==3.2
```

7.2.4. Update Config For Transfer Learning

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format

config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])

pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)

pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint =
os.path.join(paths['PRETRAINED_MODEL_PATH'],
PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'test.record')]

config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
```

7.2.5. Training the model

```
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research',
'object_detection', 'model_main_tf2.py')

command = "python {} --model_dir={} --pipeline_config_path={} --
num_train_steps=2000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
```

7.2.6. Detect from an Image

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline

category_index =
label_map_util.create_category_index_from_labelmap(files['LABELMAP'])

IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', A.02533422-940e-11eb-
9dbd-5cf3709bbcc6.jpg')

img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)
input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)
num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
          for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
label_id_offset = 1
image_np_with_detections = image_np.copy()
viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'],
        detections['detection_classes']+label_id_offset,
        detections['detection_scores'],
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=5,
        min_score_thresh=.8,
        agnostic_mode=False)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

7.2.7. Real Time Detections

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
   ret, frame = cap.read()
   image_np = np.array(frame)

   input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)
   detections = detect_fn(input_tensor)

   num_detections = int(detections.pop('num_detections'))
   detections = {key: value[0, :num_detections].numpy()
            for key, value in detections.items()}
   detections['num_detections'] = num_detections

   # detection_classes should be ints.
   detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

   label_id_offset = 1
   image_np_with_detections = image_np.copy()

   viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+label_id_offset,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.8,
            agnostic_mode=False)

   cv2.imshow('object detection',  cv2.resize(image_np_with_detections, (800, 600)))

   if cv2.waitKey(10) & 0xFF == ord('q'):
      cap.release()
      cv2.destroyAllWindows()
      break
```

CHAPTER 8

TESTING

8.1 Test Cases

Testing is a procedure for demonstrating the program's correctness. Testing is required to demonstrate completeness, improve software quality, and give maintenance assistance. As a result, some testing standards are required to reduce testing costs and time. Testing software takes place throughout the coding process and is the final check of setups, design, and coding. We can determine whether the configurations that have been developed are suitable or not based on how the software reacts to these tests. All components of an application are tested, because failing to do so leads to plenty of bugs once the software is in use. The benefits of testing include preventing bugs, reducing development costs and improving performance.

8.2 Testing Principles:

(i) All tests should meet the customer's needs.

(ii) Software testing is performed by a separate Testing Team

(iii) Exhaustive testing is not practicable. We require the optimum quantity of testing depending on the application's risk assessment.

(iv) All tests that will be undertaken should be prepared before they are carried out.

(v) It adheres to the Pareto rule (80/20 rule), which claims that 80% of software mistakes are caused by 20% of programme components.

(vi) Begin with minor parts and work your way up to larger ones.

8.3 Software testing types

There are numerous sorts of software tests, each with its own set of goals and strategies:

- Acceptance testing entails determining if the entire system functions as intended.
- Integration testing is the process of ensuring that software components or functions work in tandem.
- Unit testing is the process of ensuring that each software unit works as planned. An application's smallest testable component is called a unit.
- Testing functions by simulating business situations based on functional requirements is known as functional testing.
- Black-box testing is a typical method of ensuring that functions are working properly.

- Performance testing is the process of determining how well software performs under various workloads. For example, load testing is performed to assess performance under real-world load situations.
- Testing to see if new features fail or degrade functioning is known as regression testing.

8.4 Test Cases

Table 8. 1: Test Cases

Sr. No	Test Module	Test Case	Input Details	Expected O/p	Actual O/p	Test Status
1	Recognize Gesture Alphabet ‘A’	User should Perform the Sign Language Gesture for Alphabet ‘A’	Live Feed from Web Camera	Recognizing the Gesture as ‘A’	As Expected	Pass
2	Recognize Gesture Number ‘5’	User should Perform the Sign Language Gesture for Number ‘5’	Live Feed from Web Camera	Recognizing the Gesture as ‘5’	As Expected	Pass

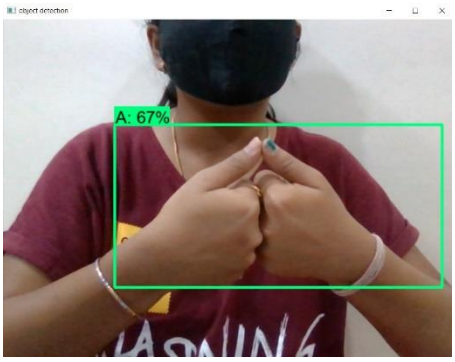

Figure 8.1: Test Case 1: Recognizing Alphabet A

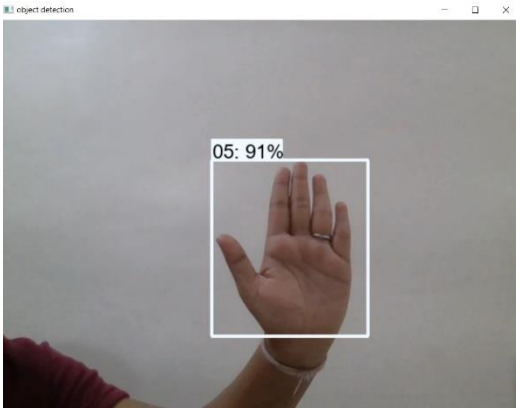

Figure 8.2: Test Case 2: Recognizing Number 05

CHAPTER 9

RESULTS & DISCUSSIONS

9.1 Experimental Results

In this part we will look at the results of the trained mobilnet v2 model. A total of 15,000 images were used in this study. The system recognizes the sign language gestures and provides us output in form of prediction of the gesture name. A large dataset is used in the framework to provide a high efficiency rate. A total of 80% of this dataset was used to train the model, while the remaining 20% is used to test it. At the end we concluded with a system with better performance and more precision. The f1 score was used to test its efficiency.

Table 9. 1: Model Performance

Model	F1 Score
MobileNet-V2 (640x640)	79.59%

- **F1 Score:**

  The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results.

  The F1-score of a classification model is calculated as :

  F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

- **Precision:**

  Precision is one indicator of a machine learning model's performance – the quality of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions. **Precision** attempts to answer the following question: What proportion of positive identifications was actually correct?

  Precision = TruePositives / (TruePositives + FalsePositives)

  For example, if the model detected 100 objects, and 90 were correct, the precision is 90 percent.

- **Recall:**

  The recall is calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples. The recall measures the model's ability to detect Positive samples. The higher the recall, the more positive samples detected. The recall cares only about how the positive samples are classified. This is independent of how the negative samples are classified

  Recall = TruePositives / (TruePositives + FalseNegatives)

  For example, if the model correctly detects 75 trees in an image, and there are actually 100 trees in the image, the recall is 75 percent.

- In object detection and classification, a model can predict a positive class or a negative class, and the predictions can be true or false. For example, when detecting the presence of an object in an image, the positive class may be "Object Present", while the negative class would be "Object Absent". A true prediction occurs when the prediction is correct, and a false prediction occurs when the prediction is incorrect. In the image besides, the red bounding boxes indicate a positive prediction, where the model predicted that there is a tree present. The dark blue bounding boxes indicate a negative prediction, where the model predicted that there is no tree present.

Figure 9.1. 1: Detection Model Results Explained

Table 9. 2:Case Description

Case No	Description	Case
1	The model predicted that there is a tree, and it is correct.	True positive
2	The model predicted that there is a tree, and it is incorrect.	False positive
3	The model predicted that there is no tree, and it is incorrect.	False negative
4	The model predicted that there is no tree, and it is correct.	True negative

- **Confusion Matrix:**

  A confusion matrix is an NxN matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. A confusion matrix for binary classification shows the four different outcomes they are: true positive, false positive, true negative, and false negative. A model is said to be the best model if diagonal values (TP, TN) of the confusion matrix are high.

Table 9. 3:Confusion Matrix

	Estimation Class		
Actual Class		**Positive**	**Negative**
	Positive	**TP**	**TN**
	Negative	**FP**	**FN**

- **Precision-recall curve:**

  This is a plot of precision (y-axis) and recall (x-axis), and it serves as an evaluation of the performance of an object detection model. The model is considered a good predictive model if the precision stays high as the recall increases.

- **Average Precision**

  Average Precision (AP) is the precision averages across all recall values between 0 and 1 at various IoU thresholds. By interpolating across all points, AP can be interpreted as the area under the curve of the precision-recall curve.

- **Intersection over Union (IoU)**

  The Intersection over Union (IoU) ratio is used as a threshold for determining whether a predicted outcome is a true positive or a false positive. IoU is the amount of overlap between the bounding box around a predicted object and the bounding box around the ground reference data.

- **Mean Average Precision**

  The Mean Average Precision (mAP) is the average AP over multiple IoU thresholds.

CHAPTER 10

CONCLUSION AND FUTURE WORK

10.1 Conclusion:

Sign language is one of the useful tools to ease the communication between the deaf and mute communities and normal society. Though sign language can be implemented to communicate, the target person must have an idea of the sign language which is not possible always. Hence our project lowers such barriers. We Implemented Recognition of 26 Alphabets and 10 Digits in our Project

10.2 Future Work:

This model can further be improvised upon by adding some more gestures and also be deployed in an Android Application or Web Application.

CHAPTER 11

APPENDIX

PCA	Principal Component analysis
CNN	Convolutional Neural Networks
GCD	Generic cosine descriptor
ISL	Indian Sign Language
3DCNN	3D convolutional neural network
MSVM	Multiclass Support Vector Machine
BID	Big Data, IoT and Data Science
ICSCET	International Conference on Smart City and Emerging Technology
ANTIC	Advanced Network Technologies and Intelligent Computing
CCIS	Communications in Computer and Information Science
IEEE	Institute of Electrical and Electronics Engineers.
OpenCV	Open Source Computer Vision Library
AP	Average Precision
IoU	Intersection over Union
mAP	Mean Average Precision

BIBILOGRAPHY AND REFERENCES

References:

[1] P.B.Patel, Suchita Dhuppe, Vaishnavi Dhaye "Smart Glove For Deaf And Dumb Patient " International Journal of Advance Research in Science and Engineering, Volume No.07, Special Issue No.03, April 2018

[2] Khushboo Kashyap, Amit Saxena, Harmeet Kaur, Abhishek Tandon, Keshav Mehrotra "Digital Text and Speech Synthesizer using Smart Glove for Deaf and Dumb" International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 6, Issue 5, May 2017

[3] Anup Kumar, Karun Thankachan and Mevin M. Dominic "Sign Language Recognition" 3rd InCI Conf. on Recent Advances in Information Technology I RAIT-20161

[4] Muthu Mariappan H, Dr Gomathi V "Real-Time Recognition of Indian Sign Language" Second International Conference on Computational Intelligence in Data Science (ICCIDS-2019)

[5] Nitin R. Gavai,Yashashree A. Jakhade,Seema A. Tribhuvan, Rashmi Bhattad "MobileNets for Flower Classification using TensorFlow" 2017 International Conference on Big Data, IoT and Data Science (BID) Vishwakarma Institute of Technology, Pune, Dec 20-22, 2017

[6] Aditya Das, Shantanu Gawde, Khyati Suratwala and Dr. Dhananjay Kalbande "Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images" Department of Computer Engineering Sardar Patel Institute of Technology Mumbai, India

[7] Neel Kamal Bhagat, Vishnusai Y,Rathna G N "Indian Sign Language Gesture Recognition using Image Processing and Deep Learning" Department of Electrical Engineering Indian Institute of Science Bengaluru, Karnataka ©2019 IEEE

[8] Divya Deora, Nikesh Bajaj "Indian Sign Language Recognition" 2012 1st International Conference on Emerging Technology Trends in Electronics, Communication and Networking ©2012 IEEE

[9] Sruthi C. J and Lijiya A "Signet: A Deep Learning based Indian Sign Language Recognition System" International Conference on Communication and Signal Processing, April 4-6, 2019, India

[10] Lean Karlo S. Tolentino, Ronnie O. Serfa Juan, August C. Thio-ac, Maria Abigail B. Pamahoy, Joni Rose R. Forteza, and Xavier Jet O. Garcia "Static Sign Language Recognition Using Deep Learning" International Journal of Machine Learning and Computing, Vol. 9, No. 6, December 2019

[11]Persons with Disabilities (Divyangjan) in India. New Delhi, India: Ministry of Statistics and Programme Implementation, Government of India, 2021.

[12]Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications ",Computer Vision and Pattern Recognition, Cornell University.

[13]LabelImg - A tool used for Image Annotation of dataset - https://github.com/tzutalin/labelImg.git

[14]Google, "MobileNets: Open-Source Models for Efficient On-Device Vision," Research Blog. [Online]. Available:https://research.googleblog.com/2017/06/mobilenets-open-source-models-for.html.

[15]https://innovate.mygov.in/

[16]https://towardsdatascience.com/sign-language-recognition-using-deep-learning-6549268c60bd

[17]https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/MobileNetV2

ACKNOWLEDGEMENT

We would like to express our deepest gratitude to all those who provided us the possibility to complete this report. We have taken efforts in this project, but it would not have been possible without the kind support and help of many individuals. We would like to extend our sincere thanks to all of them.

We take this opportunity to express our profound gratitude and deep regards to our teacher **Mrs. Brinzel Rodrigues** for her exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by her time to time shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to our Head of Information Technology Department, **Dr. Arun Saxena** along with all the Teaching & Non-Teaching Staff for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We express our sincere gratitude to our respected principal **Dr. G. V. Mulgund** for encouragement and facilities provided to us. We would also like to acknowledge the patience that our ever-beloved parents have shown during our efforts and the encouragement we have received from them.

Last but not least we would also like to thank to our Friends and Family who directly and indirectly helped and supported us during the whole course of the Project.

Amit Ravindra Kuveskar

Pranali Ratilal Wadile

Ankita Mallikarjun Dodamani